Pajek – Program for Large Network Analysis

Vladimir Batagelj and Andrej Mrvar
University of Ljubljana
{vladimir.batagelj,andrej.mrvar}@uni-lj.si

May 28, 1997 / January 3, 1999

Abstract

Large networks, having thousands of vertices and lines, can be found in many different areas, e. g: genealogies, flow graphs of programs, molecule, computer networks, transportation networks, social networks, intra/inter organisational networks ... Many standard network algorithms are very time and space consuming and therefore unsuitable for analysis of such networks. In the article we present some approaches to analysis and visualisation of large networks implemented in program **Pajek**. Some typical examples are also given.

1 Introduction

Pajek (Slovene word for Spider) is a program, for Windows (32 bit), for analysis of *large networks*. It is freely available, for noncommercial use, at its homepage:
http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Large networks can be found in many different areas. Usually they are produced automatically, using computers, from different data sources that are already available in computer readable form. For example:

- large genealogies (genealogies having some 10.000 people [21]), Theoretical Computer Science Genealogy (1.882 persons [30]);
- networks derived from dictionaries and other texts (character mutation/insertion/ deletion network on 52.652 English words [24]);
- transportation networks (American airlines with 332 airports [31]);
- large molecule (molecule having thousands of atoms, e. g. DNA [28]);
- communication networks: links among pages or servers on Internet, usage of Usenet [29], phone calls [20];

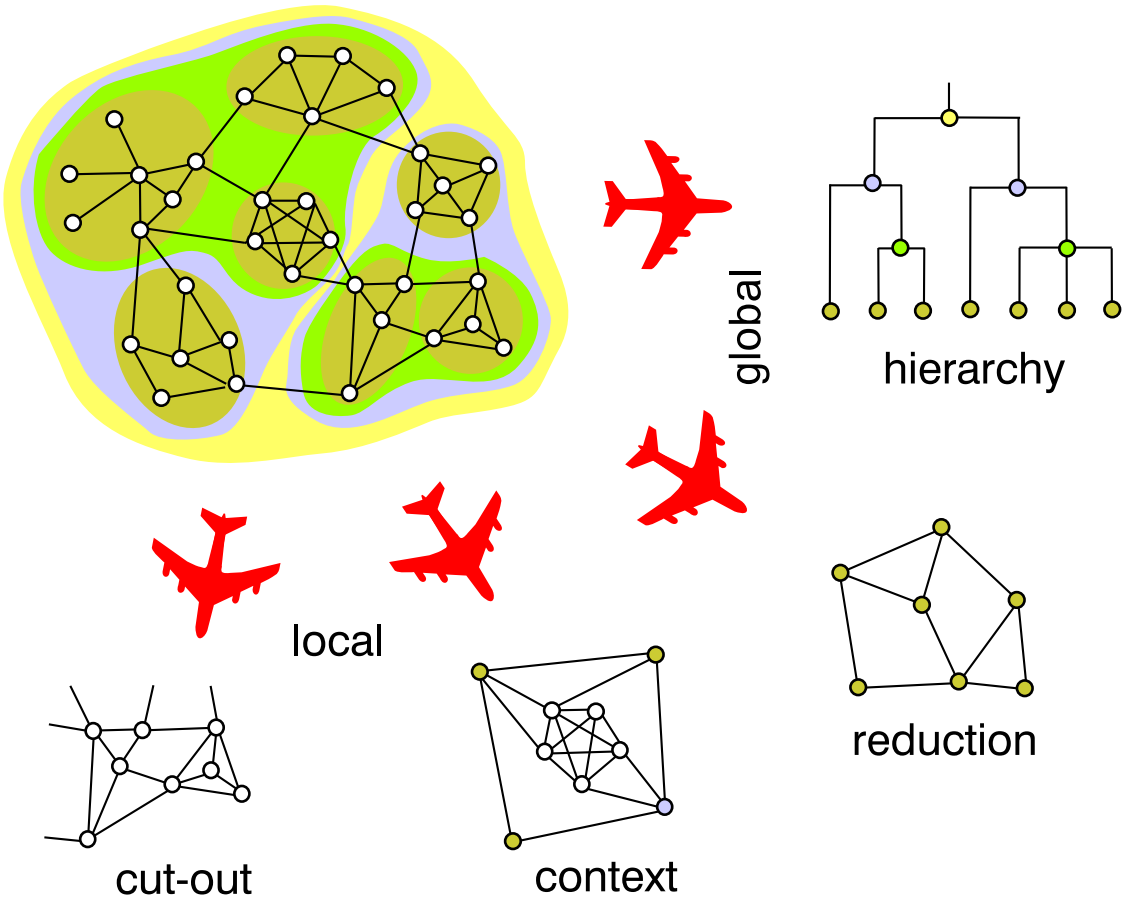

Figure 1: Goals.

- flow graphs of programs [15];
- bibliographies, citation networks [9, 7], Erdös graph (network with 5.822 co-authors [18]), ...

Such networks cannot be treated efficiently using standard network analysis tools which are mostly based on matrix representation and are therefore limited to networks of moderate size (some tens or hundreds of vertices at most).

The main goals in the design of **Pajek** are (see Figure 1):

- to support *abstraction* by (recursive) factorization of a large network into several smaller networks that can be treated further using more sophisticated methods;
- to provide the user with some powerful *visualisation* tools;
- to implement a selection of *efficient* algorithms for analysis of large networks.

One of the approaches to support abstraction is: *find* clusters (components, neighbourhoods of 'central' vertices, cores, ...) in a network, *extract* and *show* vertices that belong to the same clusters separately, possibly with the parts of the context (detailed local view), *shrink* vertices in clusters and show relations among clusters (global view).

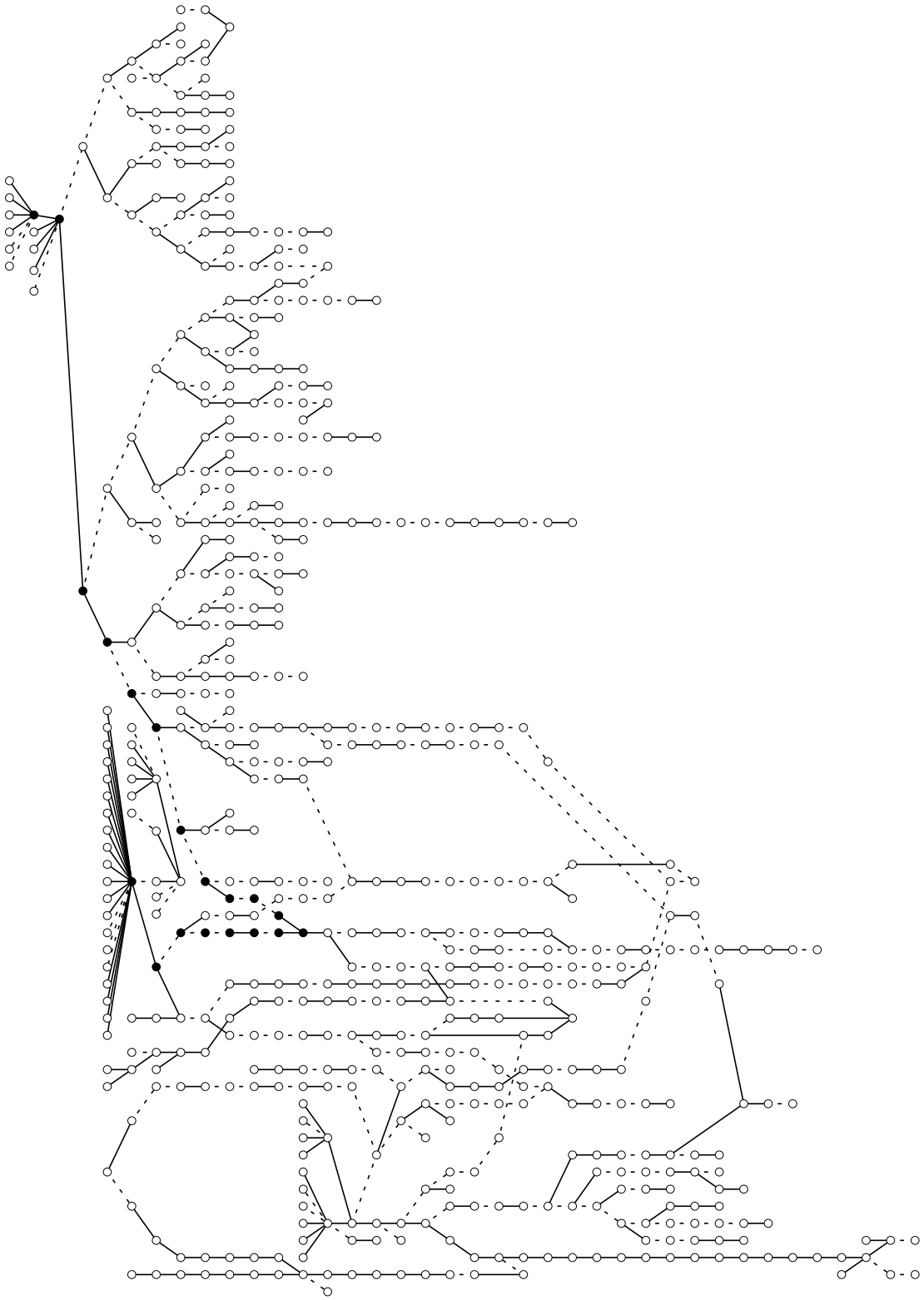

Figure 2: USA presidents genealogy.

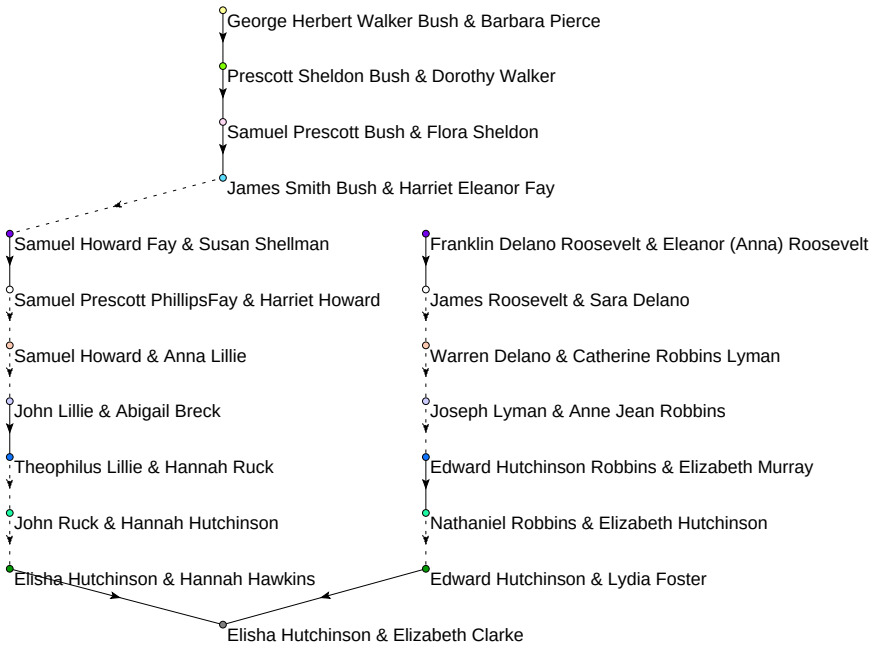

Figure 3: Shortest path between Franklin Delano Roosvelt and George Herbert Walker Bush in USA presidents genealogy.

Table 1: Time complexities of algorithms (Pentium/64M/90MHz).

	T(n)	1.000	10.000	100.000	1.000.000
Shuffle	$\mathcal{O}(n)$	0,00 s	0,015 s	0,17 s	2,22 s
Quick Sor	$\mathcal{O}(n \log n)$	0,00 s	0,00 s	0,40 s	5,14 s
Heap Sort	$\mathcal{O}(n \log n)$	0,00 s	0,06 s	0,98 s	14,35 s
Insertion Sort	$\mathcal{O}(n^2)$	0,07 s	7,50 s	`12,5` min	`20,83` hours
XY	$\mathcal{O}(n^3)$	`0,10` s	`1,67` min	`1,16` d	`3,17` years

2 Efficient algorithms for large networks

Time $T(n)$ and *space* $S(n)$ *complexities* of an algorithm are estimates of the time and memory space needed to run it on instances of size n (in our case – number of vertices or lines).

In most large networks the number of lines m is of the same order as the number of vertices – $\mathcal{O}(n)$ or at most $\mathcal{O}(n \log n)$ (such networks are considered *sparse networks*). In the following we assume that we have to analyse large but sparse networks.

According to capabilities of nowadays computers, space complexity for storing sparse networks is not crucial any more. The problem can be solved using appropriate data structures for internal representation of data (doubly linked lists representation of networks is used in Pajek).

Having much faster computers does not help a lot in the case of high order time complexities. In the theory problems solvable with algorithms of polynomial complexity are considered easy.

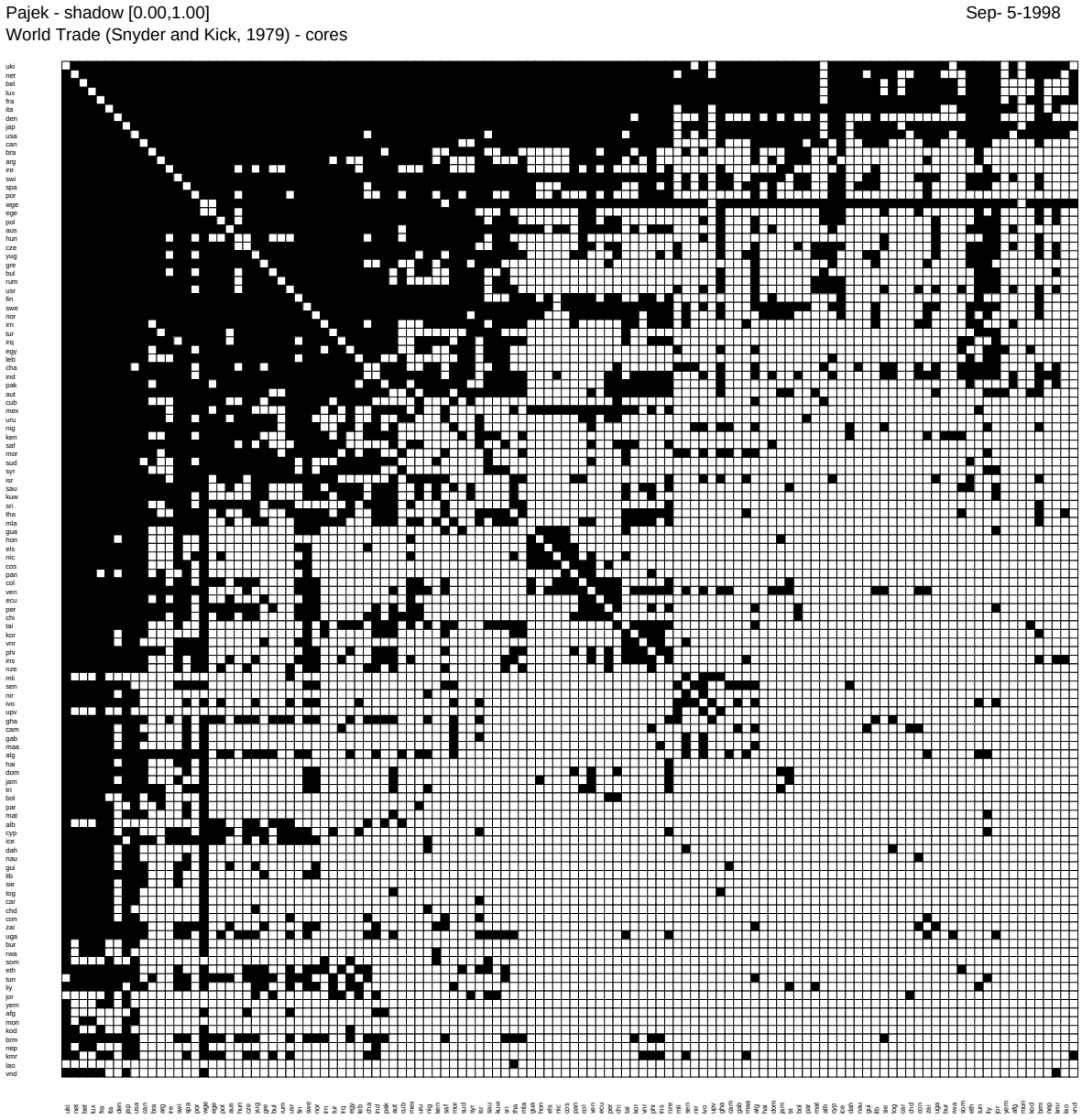

Figure 4: Reordered Snyder and Kick's world trade matrix.

But, in the case of large n, in practice already algorithms of time complexity of order $\mathcal{O}(n^2)$ can be too slow (for the interactive use), what can be seen in the Table 1.

Therefore, most of the algorithms implemented in **Pajek** have *subquadratic* time complexities: $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n\sqrt{n})$, or are restricted to small sets of selected vertices.

3 Data Structures

Currently **Pajek** uses six data structures to implement the algorithms:

- *network* – main object (vertices and lines);
- *permutation* – reordering of vertices;
- *vector* – values of vertices;
- *cluster* – subset of vertices (e. g. one class from partition);
- *partition* – tells for each vertex to which cluster the vertex belongs;
- *hierarchy* – hierarchically ordered clusters and vertices.

The power of **Pajek** is based on several transformations which support different transitions among these data structures.

Besides its own input formats, Pajek supports several other formats: `UCINET DL`; `GED` [19], genealogies can be read either as Ore-graph or p-graph [21, 16, 17]; and some molecular formats: `BS` (Ball and Stick), `MAC` (Mac Molecule) and `MOL` (MDL MOLfile).

4 Algorithms

Using the above data structures the basic set of efficient algorithms was implemented [1, 24, 13, 14], for example:

- *partitions*: degree, depth, core, p-cliques, centers;
- *binary operations*: union, intersection, difference;
- *components*: strong, weak, biconnected, symmetric [1];
- *decompositions*: symmetric-acyclic;
- *paths*: shortest path(s), all paths between two vertices [11];
- *flows*: maximum flow between two vertices [14];
- *citation weights*: Paths Count Method and SPLC Method [2];
- *neighbourhood*: k-neighbours;
- *CPM* (Critical Path Method);
- *extracting* subnetwork;
- *shrinking* clusters in network (generalized blockmodeling) [4];
- *reordering*: topological ordering, Richards's numbering, depth/breadth first search;

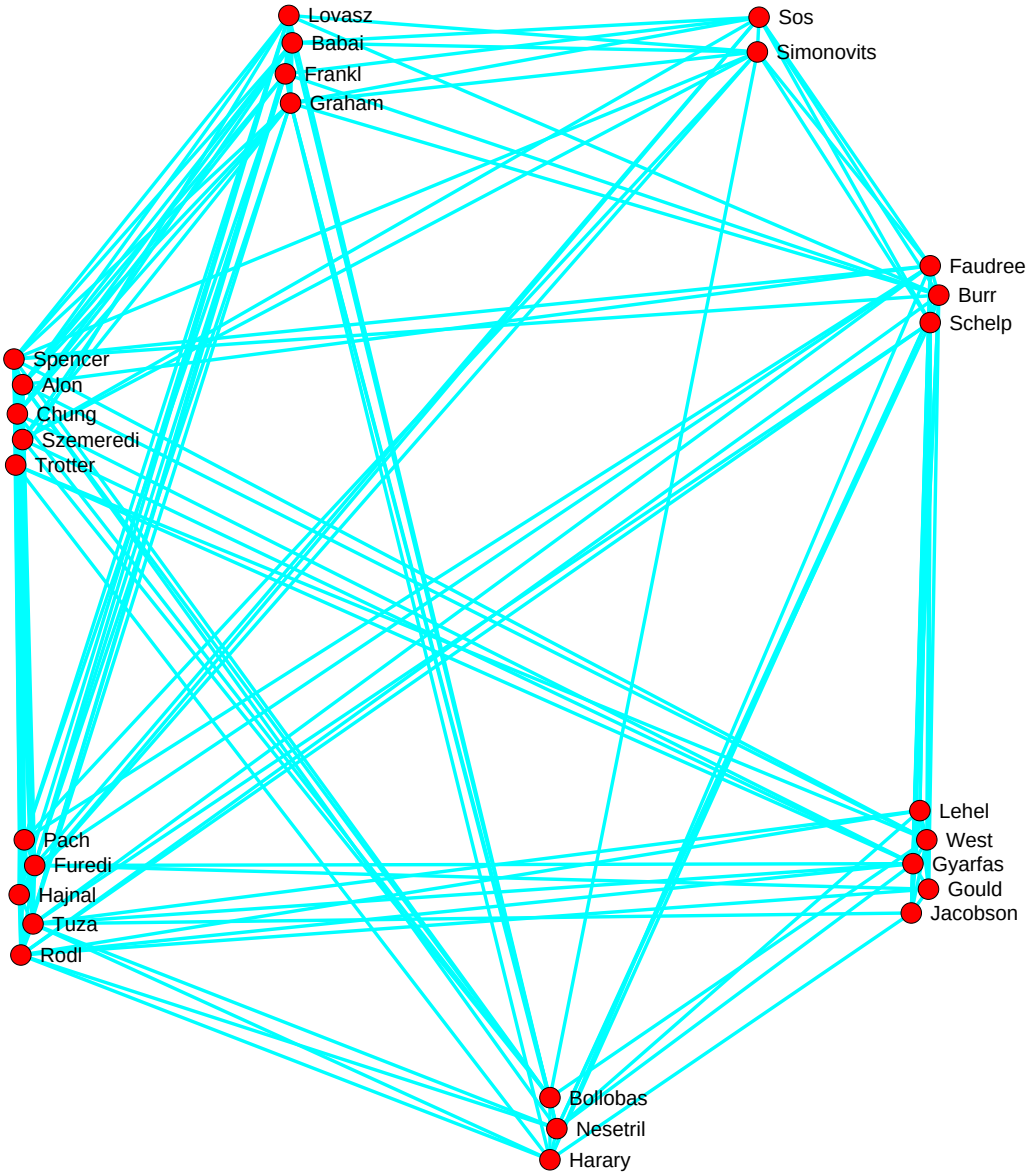

Figure 5: Clique decomposition of 9-core of Erdös graph.

- *reduction*: hierarchy, subdivision, degree;
- *simplifications and transformations*: deleting loops, multiple lines, transforming arcs to edges ...

We can define an often used sequence of elementary operations as a *macro* and run it as a single command. Using systems of macros we can adapt **Pajek** to special groups of users (analysis of genealogies, chemical applications, ...).

Some *special algorithms* for solving problems from different areas of network analysis were also included in **Pajek**: e. g. algorithms for checking whether a program is written structurally [15]; simulation of Petri nets [12]; searching for given fragments/patterns in molecule or genealogies.

Special emphasis was given to automatic generation of network *layouts*. Several standard algorithms for automatic graph drawing were implemented: spring embedders based on minimisation of the total energy of the system (Kamada-Kawai [10] and Fruchterman-Reingold [6]), layouts determined by eigenvectors (Lanczos algorithm [3, 5]), drawing in layers (genealogies and other acyclic structures).

These algorithms were modified and extended to enable additional options: drawing with constraints (optimisation of the selected part of the network, fixing some vertices to predefined positions, using values of lines as similarities or dissimilarities), drawing in space. **Pajek** also provides tools for *manual graph editing*.

Pajek supports several output graphic formats which can be examined by special 2D and 3D viewers (Encapsulated PostScript – `GSView` [25]; VRML – `CosmoPlayer` [23]; MDL-MOL – `Rasmol` [28], `Chime` [22]; Kinemages – `Mage` [26]).

5 Examples

In Figure 2 the largest connected p-graph component of the Genealogy of the USA presidents [32] is presented. The shortest kinship path between Franklin Delano Roosvelt and George Herbert Walker Bush, determined using macro Path, is displayed in Figure 3.

Figure 4 represents a reordered matrix of Snyder and Kick's world trade relation (iterative core decomposition with additional analysis of internal structure of cores).

The cliques decomposition (obtained by **MODEL2** [27]) of the main core of Erdös graph [18] is displayed in Figure 5.

The last picture (Figure 6) presents a snapshot of the 3D display by Mage [26] of the Prison network (from the `UCINET` dataset).

6 Future plans

Pajek is in a constant development. The latest version is available from its homepage. In the near future we are planning to implement the following additional options: different clustering and decomposition procedures, some statistics (triad counts), animation and presentation of the sequence of networks, output formatting, control structures in macros, planarity testing and layout...

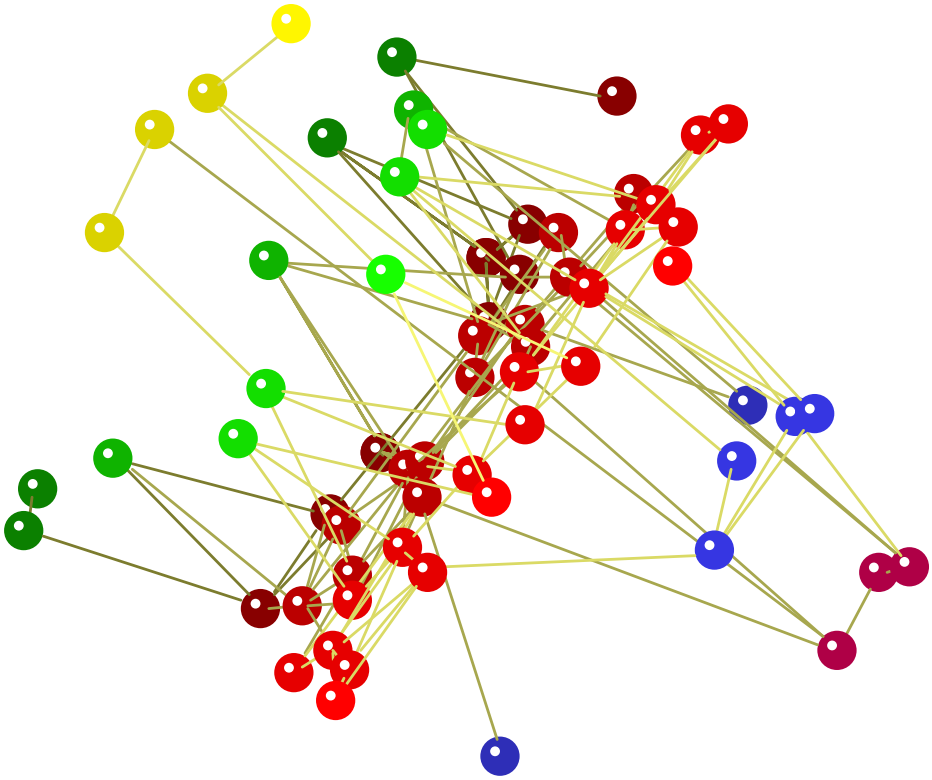

Figure 6: Prison 3D graph.

References

[1] AHO, A. V., HOPCROFT, J. E., ULLMAN, J. D. (1976): *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

[2] BATAGELJ, V. (1994): *An Efficient Algorithm for Citation Networks Analysis.* Presented at EASST'94, Budapest, Hungary, August 28-31, 1994.

[3] DATTA, B. N. (1995): *Numerical Linear Algebra and Applications.* Brooks&Cole Publishing Company, Pacific Grove.

[4] DOREIAN, P., BATAGELJ, V., FERLIGOJ, A. (1994): Partitioning Networks Based on Generalized Concepts of Equivalence. *Journal of Mathematical Sociology,* **19**, 1, 1-27.

[5] GOLUB, G. H., van LOAN, C. F. (1996): *Matrix Computations.* The John Hopkins University Press, Baltimore.

[6] FRUCHTERMAN, T. M. J., REINGOLD, E. M. (1991): Graph Drawing by Force-Directed Placement. *Software, Practice and Experience,* **21**, 1129-1164.

[7] HUMMON, N. P., DOREIAN, P. (1989): Connectivity in a Citation Network: The Development of DNA Theory. *Social Networks,* **11**, 39-63.

[8] HUMMON, N. P., DOREIAN, P. (1990): Computational Methods for Social Network Analysis. *Social Networks*, **12**, 273-288.

[9] HUMMON, N. P., DOREIAN, P., FREEMAN, L. C. (1990): Analyzing the Structure of the Centrality-Productivity Literature Created Between 1948 and 1979. *Knowledge: Creation, Diffusion, Utilization,* Vol. **11**, June, No. 4, 459-480.

[10] KAMADA, T., KAWAI, S. (1989): An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, **31**, 7-15.

[11] KNUTH, D. E. (1993): *The Stanford GraphBase.* Stanford University, ACM Press, New York.

[12] PETERSON, J. L., (1981): *Petri Net Theory and Modeling of Systems*. Prentice-Hall, Inc., Englewood Cliffs, N.J.

[13] ROGERS, E. M., KINCAID, D. L. (1981): *Communication Networks, Toward a New Paradigm for Research*. The Free Press, New York.

[14] TARJAN, R. E. (1983): *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics Philadelphia, Pennsylvania.

[15] WATSON, A. H., MCCABE, T. J. (1996): *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric.* Computer Systems Laboratory, National Institute of Standards and Technology Special Publication 500-235, Gaithersburg, MD 20899-0001.

[16] WHITE, D. R., JORION, P. (1992): Representing and Analyzing Kinship: A New Approach. *Current Athropology* **33**, 454-462.

[17] WHITE, D. R., JORION, P. (1996): Kinship Networks and Discrete Structure Theory: Applications and Implications. *Social Networks* **18**, 267-314.

[18] Erdös Number Project:
http://www.oakland.edu/~grossman/erdoshp.html

[19] GEDCOM Standard:
http://www.gendex.com/gedcom55/55gcint.htm

[20] Graph Drawing Competition 1996, Graph B:
http://portal.research.bell-labs.com/orgs/ssr/people/north/contest.html

[21] p-graphs:
http://eclectic.ss.uci.edu/~drwhite/pgraph/p-graphs.html

[22] Plug-in Chime:
http://www.mdli.com/download/chimedown.html

[23] Plug-in Cosmo Player:
http://cosmosoftware.com/

[24] KNUTH, D. E.: *Dictionary.* Stanford University, Computer Science Department:
ftp://labrea.stanford.edu/pub/dict/

[25] Program **GSView**:
ftp://ftp.cs.wisc.edu/pub/ghost/rjl/

[26] Program **Mage**:
http://www.prosci.org/Kinemage/MageSoftware.html

[27] Program **MODEL2**:
http://vlado.fmf.uni-lj.si/pub/networks/stran/default.htm

[28] Program **RasMol** (RASter MOLecules):
http://klaatu.oit.umass.edu/microbio/rasmol/getras.htm

[29] SMITH, M. A. (1996): NetScan, Department of Sociology, UCLA:
http://www.sscnet.ucla.edu/soc/csoc/netscan/netscan.htm

[30] Theoretical Computer Science Genealogy:
http://sigact.acm.org/genealogy/

[31] Transportation Networks, National Transportation Atlas Database, Bureau of Transportation Statistics:
http://www.bts.gov/gis/ntatlas/networks.html

[32] American Presidents GEDCOM file:
ftp://www.dcs.hull.ac.uk/public/genealogy/